# Deep Dive: MEANJS 1

## What is MEANJS?

- https://github.com/meanjs/mean
- Full-stack JavaScript open-source solution.
  - M: MongoDB - https://www.mongodb.com/
  - E: Express - http://expressjs.com/
  - A: Angular - http://angularjs.org/
  - N: Node.js - http://nodejs.org/

## Before we go on!

### Stack vs. Framework

**Framework**

- Provides a set of features for web/mobile applications.
- Examples:
  - Django (Python)
  - Express (Node.js)
  - AngularJS (JavaScript)
  - Symfony (PHP)

**Stack**

- A combination of different technologies.
- Tightly integrated technologies.
- Don't need to be configured independently to work together.
  - Installation is usually separate.
- Examples
  - MEAN
  - https://en.wikipedia.org/wiki/Solution_stack

### What is needed in software development?

- Idea.
- Environment.
  - Development.
  - Staging (Testing).

    - Production.
- Choosing a framework/stack.
    - What are the needs (who is it for?).
    - What is out there?
    - What do I know?
    - Is there a ready solution out there? (CMS?)
    - Does this need to be custom software?

## Development Environment.

- NOT YOUR PRODUCTION SERVER!
- Mostly your desktop/laptop
- VMs
    - General VM
    - Vagrant (https://www.vagrantup.com/)
- Virtual environments (https://github.com/ekalinin/nodeenv, https://virtualenv.pypa.io/en/stable/)
- Docker [Mostly for production] (https://www.docker.com/)

## Managing your code.

- Repositories. Why?
- Examples
    - GitHub
    - GitLab
    - BitBucket
    - Subversion
    - *Self installation.

## MVC Review

**Model**

- How is the data represented.
- Usually involves defining columns and their values.
    - model: User
        - full_name: string, 100
        - username: string, 30, not null, unique
        - email: string, 100, not null, primary key
        - password: string, 256
        - signup: timestamp
- Involves abstraction
    - Object Relational Mapping (ORM)
    - Creates an abstraction of what the database looks like.

    - Can add additional behavior to the model.
- Database Independent.
    - Underlying database can change without affecting the abstraction.
- Automation!
    - Schema created for you.
    - Queries use programming language instead of RAW queries.
- Validation
    - Validate fields based on definition.
    - Example checks.
        - Is field correct length.
        - Is field that is required filled in.
        - Is field type correct (is email entered an email).

**Views**

- Involves rendering of HTML.
- HTML TEMPLATING
- https://docs.djangoproject.com/en/1.11/ref/templates/builtins/

**API (and REST)**

- API: Application programming interface.
    - https://en.wikipedia.org/wiki/Application_programming_interface
    - Set of subroutines definitions, protocols (rules) and tools.
    - A way of interacting with a Resource that we don't necessarily have access to or control of.
    - Usually for developers.
    - Libraries.
    - Operating Systems.
    - Web APIs
        - Code Reuse.
            - Same 'data' interface for Web App and Mobile App.
        - Third party applications.
            - Add a feature that is not available.
            - Addons/Games.
            - Enhance your own applications.
        - Usually presented as a RESTful Interface.
            - Stateless!
            - Most of the time requires permission.
            - Operations determined by HTTP verbs.
                - GET
                - POST
                - PUT
                - DELETE
- Involves returning data to user.

    - JSON
    - XML
    - HTML (In which cases?)
- Several uses:
    - Front-End.
    - Single Page APPs.
    - Third Party Applications.

**Controllers**

- Business Logic.
- Database interaction.
- Authentication checks.
- Permission checks.
- Extra
    - GET/POST/COOKIES/SESSION variables.
    - Injecting user variables.

**Middleware (Addons)**

- Tools added to enhance MVC.
- Tightly integrated to the Framework.
- Example
    - HTML Rendering tag.
    - Caching scheme.
    - Permission checks.

**URLs**

- Mapping a URL to a controller.
- Clean URLs.

# Request Timeline (Board)